DECUS
PROGRAM LIBRARY

DECUS NO.	8-246
TITLE	DF32 DISK ROUTINES
AUTHOR	Garth Peterson
COMPANY	Institute of Atmospheric Sciences South Dakota School of Mines and Technology Rapid City, South Dakota
DATE	August 25, 1969
SOURCE LANGUAGE	PAL-D

DF32 DISK ROUTINES

DECUS Program Library Write-up — DECUS No. 8-246

Title: DF32 Disc Routines

Author: Garth Peterson

Date: 25 August 1969

Name: DCIO/DCNI

Core requirements: DCIO, 128 words
DCNI, 105 words

Programming language: PALD

Abstract:

DCIO and DCNI are DF32 disc input-output subroutines which operate with the interrupt facility enabled and disabled respectively. Error checking is fairly thorough and includes address checks before each disc operation and write-lock checks at the beginning and end of each disc write operation. Loading and calling of DCIO are limited to field 0; loading and calling of DCNI may be in any field, but it must be the same field.

Description:

DCIO and DCNI use the following calling sequence:

```
JMS to subroutine
  Core address
  Word count (positive)
  Low-order disc address
  UHMC, Control word
  User's error routine address (DCIO only, optional)
```

The control word breaks down thus:

U: Bits 1 and 2 specify the disc unit where transfer begins.

H: Bits 3 through 5 specify the high-order address within the disc.

M: Bits 6 through 8 specify the memory field of data transfer.

C: Bit 10 = 1 specifies external error service; bit 10 = 0 specifies an internal halt and restart after an error. Bit 11 = 1 for read, 0 for write.

A zero value of the word count argument causes a transfer of 4096 words. When DCIO is called with bit 10 on in the control word, the calling sequence must include the address of a user-written error routine which will tell DCIO what to do with an error condition. If bit 10 in the control word is off, or if DCNI is called, there are only four arguments.

Because the selection of the data transfer field is handled by the CPU and disc hardware, field restrictions on these subroutines apply only to loading and calling. DCIO must be loaded into, and called from, field zero only; the data field must also be zero when it is called. DCNI may be loaded into any field, but must be called from the same field; the data field register at the time of the call is ignored, and DCNI returns with the data field set to match the instruction field.

DCIO and DCNI both clear the accumulator and link when called. DCIO returns from the call with the accumulator and link clear, with the interrupt facility enabled, and usually with disc interrupts pending. DCNI returns with the disc not busy, with the interrupt facility disabled, and with the accumulator and link clear provided the requested operation has been successfully completed.

When an error is detected, these subroutines set up an error display consisting of the disc EMA and error bits as read by the DEAC instruction in the accumulator, together with a 1 in the link to indicate an invalid address error or with a 0 in the link for any other error. If internal error service is requested (bit 10 in the control word off), the program will halt to present the error display and will restart the disc operation after the CONTINUE key is pressed. Internal error service is not recommended for DCIO when the disc is operated concurrently with magnetic tape. Address validity, first for the terminal disc EMA and second for the initial EMA, is checked before the operation is begun. Other errors are handled postoperatively. Parity errors are retried four times automatically before any other error service, but only if there is no other error. External error service (bit 10 in the control word on) for DCNI consists of a return from the call with the error display in the accumulator and link. External error service for DCIO consists of a subroutine branch with the error display to the user's error routine. DCIO will also make this branch after a successful operation but with the accumulator and link cleared. Because this routine operates with the interrupt facility disabled and in most cases during actual interrupt service, it should be short and should terminate with an indirect branch via its entry point. After this branch the disc operation will be restarted if the accumulator is

non-zero (even if there was no error!) and will be terminated if the accumulator is clear. An example of a simple user error routine is:

```
DCERRT, 00
  DCA DCIND
  JMP I DCERRT
```

When the disc is no longer busy, location DCIND can be examined to determine whether the operation was completed or aborted.

Core location DCBSY within DCIO is negative while the disc is busy and zero when it is not busy. The calling arguments for the current operation (or previous when not busy) are present in their original forms at locations DCBSY+1 through DCBSY+5, where they may be examined or modified by the user's error routine. DCIO contains an internal subroutine, DCSERV, to which the calling program should branch during interrupt service (JMS DCSERV, effective). This branch should be unconditional; the author knows no simple and reliable way to determine whether or not a disc interrupt is present.

The invalid address checks in DCIO and DCNI are preoperative for two reasons. First, the disc address register after a read operation is set to one more than the last disc address actually read into core memory. When the last word of the last disc is read, this causes the non-existent disc flag to come on; however, it does not inhibit the completion flag, which comes on 30 microseconds later according to information from Digital Equipment Corporation. Second, if a non-existent disc flag occurs after an operation has begun, the completion flag usually does not come on; the non-existent disc flag generates an interrupt, but it is not simple to distinguish between the write-lock flag and the non-existent disc flag. While it is probably possible to write a disc subroutine to handle non-existent disc errors postoperatively, it is unlikely that the interrupt-enabled version would fit into one page of core. It is not necessary for disc 0 to be present for DCIO or DCNI to operate, but some disc must be selected in order for the interrupt facility to be enabled when DCIO is used.

Both DCIO and DCNI ordinarily clear disc flags at the end of a operation, but if an invalid address is requested and the operation is then terminated, the invalid address will remain in the disc address register and the non-existent disc flag will be on. In addition, DCIO fails to clear flags when the last word of the last disc is read. Therefore, DCSERV in DCIO has provision for finding a valid address and clearing flags while the disc is not busy. There is also a short version of DCSERV included in the disc subroutine package which only services disc interrupts while the disc is not busy and which should be included in any program that uses DCNI to operate the disc but allows interrupts elsewhere in the program.

These subroutines have several limitations. First, the handling of timing errors is not completely satisfactory. It appears that a timing

error caused by a lost data break can let the disc address register get ahead of the word count in core, causing destruction of disc data during a write operation, and possibly letting the completion flag get lost because of invalid addressing. DCSERV in DCIO ignores all interrupts while the disc is busy until the completion flag comes on: conceivably the program could hang up. Second, if disc unit select switches are changed while the disc control is busy, the completion flag could be lost. Third, there is no provision for dealing with non-existent data transfer memory fields. Fourth, the write-lock flag is checked immediately after a disc operation begins and again after it is complete, but if a write operation begins at a locked-out address and ends at an unlocked address, a timing error will cause the write-lock error indication to be lost. Fifth, these DF32 subroutines were developed on a single disc system, and the author does not know what happens if two discs are given the same unit number.